LAUREA AMMATTIKORKEAKOULU Uuden edellä

Partikkeli-editorin käytettävyys ja toteutus - Case Rovio Entertainment Oy

Ahola, Santtu

2013 Leppävaara

Laurea-ammattikorkeakoulu
Laurea Leppävaara

Partikkeli-editorin käytettävyys ja toteutus - Case Rovio Entertainment Oy

Ahola, Santtu
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Toukokuu, 2013

Laurea-ammattikorkeakoulu **Tiivistelmä**
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma

Ahola, Santtu

Partikkeli-editorin käytettävyys ja toteutus - Case Rovio Entertainment Oy

Vuosi 2013 Sivumäärä 32

Tämän opinnäytetyön tavoitteena oli tutkia partikkeli-editorien käytettävyyttä ja kerätyn tiedon avulla luoda Angry Birds Classic -pelin graafikoille työkalu partikkeli-efektien hallitsemiseen. Tarkoituksena oli luoda mahdollisimman helppokäyttöinen editori, jonka avulla graafikot kykenevät ilman ohjelmoijien tukea luomaan ja ylläpitämään partikkeli-efektejä.

Työ toteutettiin Rovio Entertainment Oy:lle ja erityisesti Angry Birds Classic -pelin tarpeet huomioon ottaen. Kyseisellä peliprojektilla ei aiemmin ollut partikkeli-editoria käytössään, joten toteutuksessa ainoana rajoitteena oli yhteensopivuus jo olemassa olevan partikkelijärjestelmän kanssa. Editori toteutettiin pelin sisäisenä osana erillisen ohjelmiston sijasta, jotta partikkeli-efektien konteksti saatiin pysymään mahdollisimman oikeana.

Työn tietopohja kerättiin benchmarking-tutkimuksella sekä käyttäjien teemahaastatteluilla. Benchmarking-osuudessa vertailtiin neljän jo olemassa olevan partikkeli-editorin käyttöliittymiä sekä ominaisuuksia. Verratut editorit olivat Particle Designer, Visionaire Studio, Unity sekä Blender, jotka kaikki palvelevat ainakin osittain erilaisia käyttötarkoituksia. Haastatteluissa kartoitettiin kolmen graafikon tarpeita ja toiveita editorin suhteen.

Näiden tietojen pohjalta kehitettiin helppokäyttöinen partikkeli-editori, jonka käyttäminen ei vaadi syvää ymmärrystä alla olevasta partikkelijärjestelmästä. Editorin avulla pelin graafinen ilme saadaan paremmin graafikoiden hallintaan ja visio partikkeli-efektien luonteesta pysyy yhtenäisempänä koko luomisprosessin ajan.

Asiasanat partikkelit, käytettävyys, pelikehitys

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Bachelor's Programme in Business Information Technology

Ahola, Santtu

Particle editor usability and implementation - a case study of Rovio Entertainment Ltd.

Year 2013 Pages 32

The purpose of this thesis was to conduct research on the usability of particle editors and based on the results to create an editor for controlling particle effects. The objective was to create a user-friendly editor that allows game artists to create and adjust particle effects without any extensive support from the programming team.

The editor was created for Rovio Entertainment Ltd. and it was designed especially with the needs of the Angry Birds Classic team in mind. The team did not previously have a particle editor available, so the only initial requirement for the new editor was compatibility with the underlying particle system. The editor was created as an internal part of the game instead of a standalone application to preserve the correct context of the particle effects.

The research section of the thesis is divided into two parts: a benchmarking study and focused interviews. The benchmarking study was used to compare the user interfaces and features of four particle editors. The chosen editors were Particle Designer, Visionaire Studio, Unity and Blender, which are all traditionally used for slightly different purposes. The interviews were focused on collecting requirements for the editor and were conducted with three game artists.

Based on the research, a user-friendly particle editor was created that does not require in-depth understanding of the underlying particle system. This allows game artists to have more extensive control over the visual style of the game and ensures that the vision over particle effects stay consistent throughout the creation process.

Keywords particles, usability, game development

Sisällys

1 Johdanto

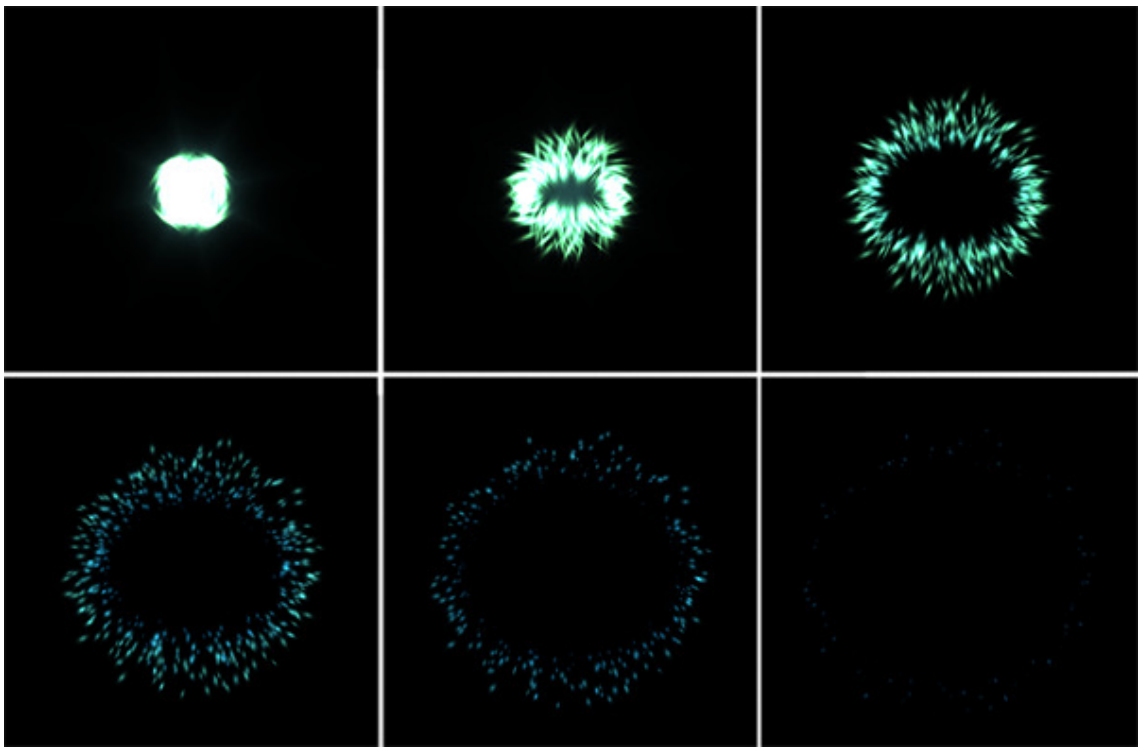
Pelien graafisessa ilmeessä partikkeli-efektit näyttelevät usein tärkeää roolia pelimaailman elävöittämisessä. Näiden efektien määrittäminen vaatii kuitenkin laajan parametri-kirjon hienovaraista tasapainottamista, joka ilman sopivia työkaluja lankeaa helposti ohjelmoijan tehtäväksi. Partikkeli-efektit ovat kuitenkin luonteeltaan hyvin graafisia, joten vision säilymisen kannalta olisi tärkeää, että graafikot kykenisivät luomaan efektejä alusta loppuun itsenäisesti. Partikkeli-editori pyrkii mahdollistamaan tämän tarjoamalla graafisen rajapinnan käyttäjän ja efektin välille, mahdollistaen täydellisen luovan kontrollin siirtämisen graafikoille.

Tämän opinnäytetyön tarkoituksena on kartoittaa tärkeimpiä osa-alueita partikkeli-editorien käytettävyydessä ja tämän kerätyn tiedon pohjalta kehittää Angry Birds Classic -pelin graafikoille nykyistä järjestelmää havainnollisempi työkalu partikkeli-efektien luomiseen ja ylläpitoon. Työ jakautuu teoria-, tiedonkeruu- sekä toteutusvaiheeseen, joista tiedonkeruuosuus jakautuu vielä benchmarking-tutkimukseen ja teemahaastatteluihin. Teoriaosuudessa kartoitetaan editori-ympäristöissä hyödyllisiä käytettävyyden alueita ja kootaan lista tärkeimmistä käytettävyyden periaatteista, joita käytetään pohjana työn muissa osuuksissa. Benchmarking-osuus vertailee jo olemassa olevien partikkeli-editorien käytettävyyttä ja kerää yhteen parhaita keinoja täyttää teoriaosuudessa kerättyjen periaatteiden kriteerit. Teemahaastattelut kartoittavat tämän työn toteutusvaiheessa kehitettävän editorin loppukäyttäjien tarpeita ja toiveita, ottaen samalla loppukäyttäjän mukaan prosessiin mahdollisimman aikaisessa vaiheessa. Työn toteutusvaiheessa luodaan suunnitelma editorin vaatimista ominaisuuksista soveltamalla aiemmissa osuuksissa kerättyjä tietoja. Tämän jälkeen käydään läpi itse editorin toteutus ja siinä kohdatut haasteet.

Vaikka tämän työn toteutusvaihe keskittyykin erityisesti Angry Birds Classic -pelin tarpeisiin, on sen yleisten havaintojen tarkoitus olla hyödyllisiä laajemminkin vastaavissa olosuhteissa. Suurin osa käytettävyyden näkökulmista partikkeli-editorien yhteydessä toimivat myös muissa editori-ympäristöissä, joten ainakin tältä osin tulokset voivat olla hyödyllisiä myös muissa projekteissa.

2 Työn tausta ja lähtökohdat

Partikkeli-efektit ovat tietokonegrafiikassa käytetty tekniikka, joka mahdollistaa esimerkiksi tulen ja sateen luonnollisen mallintamisen. Tällaiset luonnostaan kaoottiset ilmiöt ovat usein liian epämääräisiä etukäteen määriteltäviksi, joten partikkeli-efekti jakaa ilmiön yksittäisen elementin sijaan ryhmäksi itsenäisiä partikkeleita (Kuva 1). Jokainen näistä partikkeleista mallintaa tilanteeseen sopivaa yksikköä efektissä. Esimerkiksi sade-efektissä tämä yksikkö voi olla yksittäinen pisara ja räjähdyksessä vaikkapa sirpale tai kipinä. Partikkelijärjestelmä keskittää näiden yksiköiden hallinnan yhteisiin parametreihin, joiden pohjalta erillinen emitter-objekti luo uusia partikkeleita. Keskitetyn kontrollin avulla kymmenien tai jopa satojen samanaikaisten partikkelien hallinta tulee mahdolliseksi. Yleisesti tarjolla olevia parametreja ovat muun muassa partikkelien määrä, koko, elinaika sekä erilaiset partikkelin elinkaareen vaikuttavat voimat kuten tuuli tai painovoima. Näihin parametreihin liitetään usein mukaan myös hallittu määrä satunnaisuutta, jotta esimerkiksi sade-efektissä pisaroiden koot ja sijainnit saadaan jakautumaan luonnollisesti.



Kuva 1 Esimerkki partikkeli-efektin vaiheista

Partikkelijärjestelmien kirjattu historia ulottuu aina vuoteen 1982 asti, jolloin Star Trek II: The Wrath of Khan -pelin parissa työskennellyt William Reeves etsi keinoa mallintaa realistista tulta pelissään. Perinteiset mallinnustekniikat toimivat kuitenkin parhaiten tarkasti määriteltyjen objektien kanssa, joten epämääräiset efektit kuten tuli eivät sopineet tähän muottiin. Aiemmin partikkeleita oli jo käytetty muun muassa galaksien tähtien luomiseen, mutta niiden hallinta oli liian vaikeaa järjestelmällisen efektin luomiseen. Reeves keksikin yhdistää partikkeleihin sääntöjärjestelmän, jolloin saatiin aikaan kaoottinen, mutta samalla hallittu efekti. (Lander, J, 1998.)

Partikkeli-editorit tarjoavat graafisen käyttöliittymän partikkeli-efektien parametrien säätämiseen, antaen samalla välitöntä palautetta säätöjen vaikutuksista efektiin. Erillisten partikkeli-editoriohjelmistojen lisäksi vastaavia editoreja löytyy myös perinteisesti useimmista editoripohjaisista pelimoottoreista sekä monista 3D-mallinnusohjelmistoista. Nämä editorit ovat yleisesti suunnattu graafikoille tai pelisuunnittelijoille ohjelmoijien sijaan, joten niiden intuitiivisuus ja selkeys käytettäessä ovat tärkeässä roolissa työkalun hyödyllisyyttä arvioitaessa.

Tämän työn tavoitteena on tutkia partikkeli-editorien käytettävyyttä ja kerätyn tiedon avulla luoda Angry Birds Classic -pelin graafikoille työkalu näiden efektien hallitsemiseen. Aiemmin partikkelien hallintaan käytetty järjestelmä vaatii huomattavaa työpanosta myös ohjelmoijilta, joka rajoittaa graafikoiden mahdollisuuksia hallita efektien ulkoasua täysivaltaisesti. Tuotettavan partikkeli-editorin tarkoituksena on pienentää ohjelmoijien osuutta tässä prosessissa ja täten siirtää vastuuta pelin graafisesta ilmeestä graafikoiden suuntaan.

Varsinaista partikkelijärjestelmän luominen ei ole tämän työn kohteena, vaan tavoite on luoda nykyisen järjestelmän ja graafikoiden välille helppokäyttöinen rajapinta. Odotettavissa on, että nykyinen partikkelijärjestelmä tullaan korvaamaan uudella ja monipuolisemmalla järjestelmällä tulevaisuudessa, joten tämän muutoksen huomioon ottaminen on myös tärkeää tuotoksen hyödyllisyyden varmistamiseksi myös tulevaisuudessa. Editorin tulee siis palvella tämän hetken järjestelmää mahdollisimman tehokkaasti, mutta sen laajentamisenkin täytyy olla helppoa ja joustavaa. Tavoitteena on myös välttää huomattavia muutoksia nykyiseen järjestelmään, sillä nämä muutokset vaikuttavat myös muihin projekteihin.

2.1 Kohdeyrityksen esittely

Tämä opinnäytetyö toteutettiin Rovio Entertainment Oy:lle. Alunperin Relude nimellä vuonna 2003 perustettu Rovio on suomalainen pelialan yritys, joka menestyksensä myötä on laajentanut toimintaansa myös muille viihteen aloille. Yrityksen toiminta kattaakin nykyään pelien lisäksi myös muun muassa animaatioiden tuotantoa sekä kirjojen kustannustoimintaa. Rovio myös lisensoi brändejään aktiivisesti ulkoisille toimijoille.

Vuoden 2009 joulukuussa julkaistu Angry Birds oli Rovion läpimurtohitti ja kyseinen brändi onkin pysynyt yrityksen vahvan kasvun kulmakivenä viime vuosien aikana. Toukokuussa 2012 Angry Birds -pelejä oli ladattu jo yli miljardi kertaa ja saman vuoden joulukuun aikana Rovion peleillä oli 263 miljoonaa pelaajaa (263 million monthly active users... 2013). Tätä suurta käyttäjäkuntaa Rovio on hyödyntänyt esimerkiksi julkaisemalla Angry Birds Toons -animaatiosarjan paitsi tavallisten kanavien kautta, myös peleihin integroidun video-palvelun välityksellä. Vastaavien toimien myötä yhä suurempi osa Rovion tuotteista ja palveluista on koko laajeneva käyttäjäkunnan saatavilla.

Vuoden 2012 aikana Rovion henkilöstön määrä kaksinkertaistui ja vuoden lopussa Rovio työllistikin jo yli 500 henkeä, joista suurin osa työskentelee Suomessa (Lappalainen 2013). Toimintaa yhtiöllä löytyy kuitenkin myös Kiinasta ja Ruotsista. Henkilöstön määrä on kasvanut vahvasti viimevuosina paitsi rekrytoinnin, myös yrityskauppojen avulla. Vuoden 2011 puolivälissä Rovio osti helsinkiläisen animaatiostudio Kombon (Vaalisto 2011) ja seuraavana vuonna kohteena oli Futuremark-yhtiön peliosasto (Rovio teki yrityskaupan... 2012).

Rovion liikevaihto vuonna 2012 oli 152 miljoonaa euroa, josta tulokseksi jäi 77 miljoonaa euroa ennen veroja. Vuodesta 2011 kasvua liikevaihdossa oli 101 prosenttia ja tuloksessa 64 prosenttia. Oheistuotemyynnin osuus liikevaihdosta ylsi 45 prosenttiin, joka tarkoitti kolminkertaistumista kuluttajatuotemyynnissä edellisvuodesta. Yhtiön tulevaisuuden näkymät riippuvat pitkälti uusien pelien julkaisuista sekä uuden sisällön kyvystä pitää fanit sitoutuneina Rovion tuotteisiin ja palveluihin. (Lappalainen 2013.)

2.2 Työn rakenne

Opinnäytetyö jakaantuu kolmeen pääosaan: käytettävyyden teoria, tiedonkeruu sekä varsinainen partikkeli-editorin toteutus. Teoria osuus käy läpi tärkeimpiä käytettävyyden periaatteita, joita käytetään pohjana tiedonkeruuvaiheessa sekä toteutuksessa. Toteutusvaihe kartoittaa vielä lopuksi toteutetun editorin ominaisuudet sekä kohdatut haasteet.

Teoria-osuus koostuu yhdeksästä pääperiaatteesta, jotka koettiin erityisen tärkeiksi editori-ympäristöissä. Tiedonkeruuvaiheessa verrataan neljää jo olemassa olevaa partikkeli-editoria (Particle Designer, Visionaire Studio, Unity sekä Blender) ja pyritään kartoittamaan parhaita keinoja täyttää teoriaosuudessa kerätyt periaatteet. Lisäksi tiedonkeruuosuudessa käydään läpi myös kolmen graafikon teemahaastattelut, joiden tarkoituksena on luoda tarkempaa näkemystä siihen, mitkä ovat loppukäyttäjän kannalta tärkeimpiä huomion kohteita. Lopussa käydään vielä läpi varsinainen toteutus, siinä kohdatut haasteet sekä projektin onnistumisen arviointi antamalla editori loppukäyttäjien testattavaksi.

3 Käytettävyyden teoriaa

Partikkeli-ediorit sisältävät suuren määrän vaikeasti visualisoitavia säätöjä, mutta niihin pätevät silti hyvin käytettävyyden perusperiaatteet. Tämän työ käyttää pohjana käytettävyyden teorialle Jakob Nielsenin ja Ben Shneidermanin kirjoituksia aiheesta. Kummatkin ovat pitkän linjan käytettävyyden asiantuntijoita, jotka ovat kirjoittaneet useita käytettävyyden perusteoksista.

Pääosin Nielsenin ja Scheidermanin kirjoitusten pohjalta kerättiin lista tärkeimmistä periaatteista partikkeli-editorien käytettävyydestä, jota hyödynnetään kaikissa työn vaiheissa. Ensin benchmarking-tutkimuksessa niitä käytetään fokusoimaan muiden editorien toimintoja ja luomaan yhtenäinen rakenne tutkimukselle. Haastattelu- ja toteutusvaiheessa nämä periaatteet ohjaavat alueita, joihin kiinnitetään erityisesti huomiota. Tämä listaus periaatteista ei keskity yksinään mihinkään yhteen lähteeseen, vaan pyrkii olemaan kattava listaus hyvin editori-ympäristöön sopivista periaatteista. Nämä kerätyt periaatteet on listattu alla ja ne löytyvät tiivistetysti taulukosta 1.

Erilläänkin olevien toimintojen tulisi käyttäytyä johdonmukaisesti. Myös käytetyn värityksen, sanaston sekä elementtien asettelun tulisi säilyä yhtenäisenä. Tällä vähennetään käyttäjän turhauttamista ja tehostetaan toimintojen oppimista. (Shneiderman 1998, 74.)

Kokeneille käyttäjille tulisi tarjota mahdollisuus käyttää pikanäppäimiä ja muita oikoteitä. Kun käyttäjä on kerännyt riittävästi kokemusta ohjelmiston parissa, haluaa hän todennäköisesti siirtyä tehokkaampiin toimintoihin, jotka kuitenkin vaativat pohjalle paremman ymmärryksen toimintojen luonteesta. Näin voidaan vähentää vaadittujen toimien määrää halutun lopputuloksen saavuttamiseksi ja nopeuttaa työskentelyä. (Shneiderman 1998, 74.)

Vahvasti kokeiluun ja nopeaan iterointiin perustuva suunnittelutyö vaatii käyttöliittymän, joka antaa välitöntä palautetta käyttäjän toimille (Shneiderman 1998, 74-75). Välitön palaute myös vähentää virheilmoitusten tarvetta, sillä ongelmatilanteet ilmenevät välittömästi jo muutosta tehtäessä (Shneiderman 1998, 228). Näin saadaan aikaiseksi paitsi tehokkaampi työympäristö, myös mahdollisuus löytää uusia ideoita "onnellisten vahinkojen" ja yllättävien kokeilujen kautta, jotka jäisivät hitaammalla järjestelmällä kokeilematta (Victor 2012).

Toimintojen tulisi ryhmittyä loogisiksi kokonaisuuksiksi, jotka käyttäjä voi suorittaa erikseen ja ymmärtää niiden vaikutuksen kokonaisuuteen. Näin luodaan käyttäjälle luotto siitä, että kaikki tarvittavat säädöt on lopuksi käyty läpi. Samalla saadaan käyttäjälle tunne toimintojen loogisesta etenemisestä ja annetaan vihje siitä, mitkä tiedot ovat sillä hetkellä pidettävä muistissa. (Shneiderman 1998, 75.)

Käyttäjän ei tulisi pystyä tekemään vakavia virheitä, jotka vaarantavat järjestelmän vakauden. Niissä tapauksissa, joissa virheitä voi tapahtua, tulisi käyttäjälle tarjota selkeät ohjeet tilanteen korjaamiseen eikä käyttäjän tulisi joutua korjaamaan muuta kuin virheellinen osuus. Virheiden tulisi myös jättää järjestelmä alkuperäiseen tilaan, eikä antaa virheiden vaikuttaa toimintaan. (Shneiderman 1998, 75.)

Käyttäjän tulisi hallita toimintoja täysivaltaisesti, eikä hänen tulisi joutua reagoivaan rooliin. Kaikkien kokonaisuuteen vaikuttavien toimintojen ja tarvittavan tiedon pitäisi olla käyttäjän ulottuvilla, jottei käyttökokemus muutu turhauttavaksi. Yllättävät toimintojen vaikutukset sekä toimintojen hankalat toteutukset lisäävät tätä turhautumista. (Shneiderman 1998, 75.)

Käyttäjän muistia ei tulisi rasittaa turhaan, vaan tarvittavan tiedon tulisi olla aina helposti saatavilla, vaikka se olisi käyttäjän itsensä asettamaa (Shneiderman 1998, 75). On kuitenkin otettava huomioon, ettei epäolennainen tai harvoin tarvittu tieto päädy kilpailemaan tärkeämmän tiedon kanssa (Nielsen 1995).

Käyttäjän tulisi pystyä peruuttamaan toimintojaan helposti. Tämä luo turvallisen ympäristön toimintojen kokeilemiselle ja vähentää virheiden pelkoa, joka puolestaan lisää luovan kokeilun mahdollisuuksia. Peruutettava yksikkö voi olla yksittäinen toiminto tai kokonainen ryhmä toimintoja. (Shneiderman 1998, 75.)

Vaikka käyttöliittymä tulisikin ensisijaisesti olla käytettävissä ilman erillistä dokumentaatiota, on eri toimintojen dokumentointi suositeltavaa. Tällaisen ohjeistuksen tulisi olla mahdollisimman konkreettista ja käyttäjän tarpeisiin keskittynyttä. Myös helppo selattavuus sekä tiiviys ovat tärkeitä ohjeistukselle. (Nielsen 1995.)

Periaate	Lähteet
Toimintojen johdonmukaisuus	Shneiderman (1998)
Tehokäyttäjien toiminnot (pikanäppäimet)	Shneiderman (1998)
Välitön palaute	Victor (2012), Shneiderman (1998)
Toimintojen tehokas ryhmittely	Shneiderman (1998)
Virhetilanteiden riittävä hallinta	Shneiderman (1998)
Käyttäjän täyden kontrollin varmistaminen	Shneiderman (1998)
Käyttäjän muistin rasittamisen välttäminen	Nielsen (1995), Shneiderman (1998)
Mahdollisuus toimintojen peruuttamiseen	Shneiderman (1998)
Riittävä dokumentaatio	Nielsen (1995)

Taulukko 1 Kerätyt periaatteet tiivistettynä

4 Tutkimuksen toteutus ja tulokset

Tämän työn tiedonkeruuosuus jakautuu kahteen osioon: olemassa olevien editorien vertailuun benchmarking-tutkimuksen avulla sekä tulevien käyttäjien teemahaastatteluihin. Benchmarking-tutkimuksen tarkoituksena on luoda pohja uudelle editorille jo käytössä olevien parhaiden toteutusten perusteella. Tämän pohjan kehittämistä ja ennen kaikkea erikoistumista jatketaan teemahaastatteluilla, jotka ohjaavat lopullisen editorin toteutusta.

Koska useassa pelimoottorissa on jo olemassa oma partikkeli-editorinsa ja markkinoilta löytyy myös useampia itsenäisiä partikkelien suunnitteluun tarkoitettuja ohjelmia, on näiden editorien benchmarking-vertailu luonnollinen ensimmäinen askel käytettävyyden tutkinnassa. Tämän vertailun avulla luodaan pohja jo olemassa olevista ratkaisuista ja säästytään pyörän uudelleen keksimiseltä. Verrattaviksi editoreiksi tähän työhön valikoituivat Particle Designer, Visionaire Studio, Unity sekä Blender, joiden perustiedot on listattu alla (Taulukko 2).

Nimi	Versio	Kehittäjä	Tyyppi	Kotisivu
Particle Designer	1.3.6	71 Squared	Partikkeli-editori	71squared.com
Visionaire Studio	3.7.1	Visionaire Team	Pelimoottori	visionaire-studio.net
Unity	4.1	Unity Technologies	Pelimoottori	unity3d.com
Blender	2.65a	The Blender Foundation	3D-mallinnusohjelma	blender.org

Taulukko 2: Verratut editorit

Teemahaastatteluiden tarkoituksena oli ottaa loppukäyttäjä mukaan editorin kehitysprosessiin mahdollisimman aikaisin ja tarjota mahdollisuus ilmaista toiveita toteutettavan editorin suhteen. Samalla kartoitettiin aiempia kokemuksia muista vastaavista editori-ympäristöistä. Näiden haastatteluiden avulla editorin suunnittelu pidettiin mahdollisimman käytännönläheisenä.

4.1 Benchmarking suunnittelu

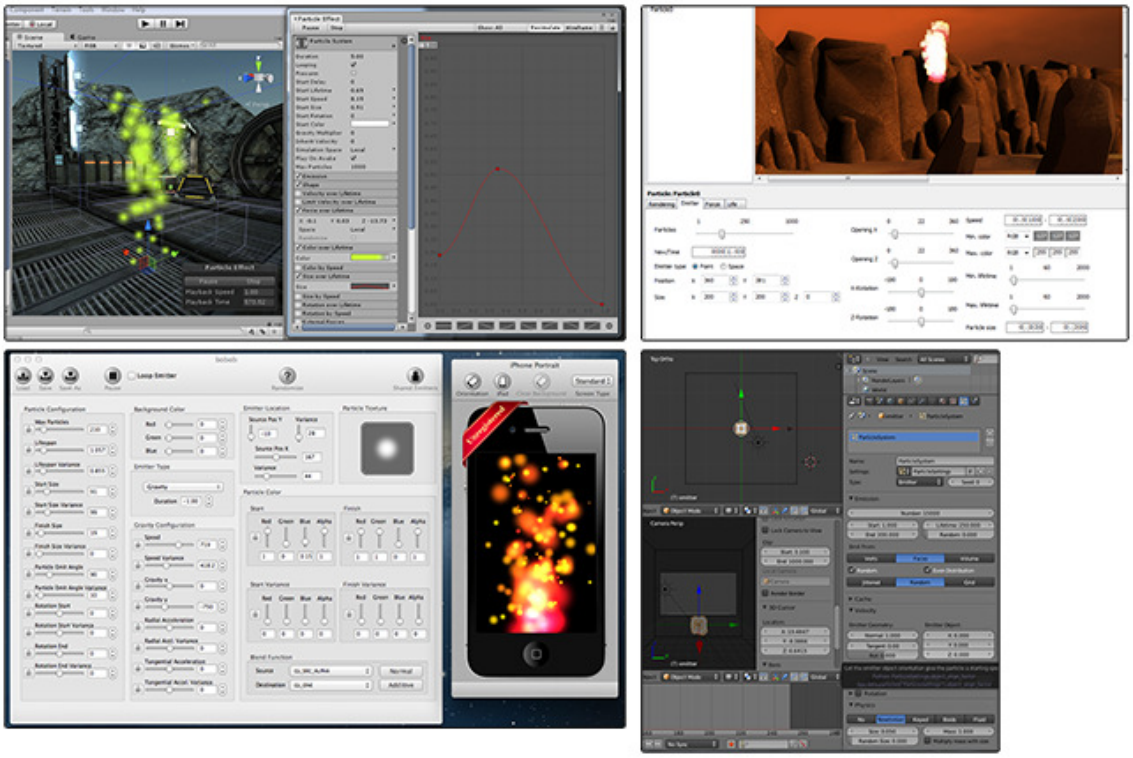
Benchmarking-vertailuun valittiin neljä partikkeli-editoria, joista kaikki palvelevat ainakin osittain erilaisia käyttötarpeita. Vaikka kaikki näistä editoreista eivät vastaakaan täysin samaa käyttötarkoitusta kuin työn lopussa toteutettava editori, niin ottamalla mukaan mahdollisimman paljon toisistaan erottuvia partikkeli-editoreja, saatiin pienellä otannalla laaja katsaus erilaisten ympäristöjen hyvistä ja huonoista puolista. Valittujen editorien kuvaukset käydään läpi alla.

Particle Designer on erityisesti iOS laitteita varten kehitetty helppokäyttöinen partikkeli-editori. Sen tarkoituksena on mahdollistaa efektien luominen ilman suurempaa perehtymistä asiaan ja tarjota siten aloittelijaystävällinen ympäristö partikkelien hallintaan. Se on tutkimuksen ainut itsenäinen, pelkästään partikkeli-efektien luomiseen keskittynyt ohjelmisto.

Visionaire Studio on pelimoottori ja editori, joka pyrkii tarjoamaan mahdollisimman yksinkertaisen väylän seikkailupelien kehittämiseen. Sen partikkeli-editori on hyvin yksinkertainen ja rajoittunut, sillä se pyrkii olemaan myös hyvin aloittelijaystävällinen. Rajoituksistaan huolimatta sillä on myös etunsa muun muassa dokumentaation laadussa.

Myös Unity kuuluu pelimoottori ja editori -kategoriaan. Se on yksi tämän hetken näkyvimmistä kaupallisista pelimoottoreista helppokäyttöisen editorinsa ja laajan laitetukensa ansiosta. Sen partikkeli-editori on huomattavasti vertailun toista pelimoottori-editori yhdistelmää, Visionaire Studiota, monipuolisempi ja kypsempi.

Blender on 3D-mallinnukseen käytettävä avoimenlähdekoodin ohjelmisto. Vaikka sen kohteena onkin lähinnä 3D-mallinnus, on sen käyttöliittymää uudistettu vahvasti viimeisen vuoden aikana, joka tekee siitä hyvän kohteen tutkimukselle. Blender tukee myös erilaisia partikkeli-efektien erikoistapauksia, kuten karvoituksen mallinnusta, mutta näihin ei keskitytä tässä tutkimuksessa.



Kuva 2 Editorien käyttöliittymät (Ylhäällä: Unity ja Visionaire Studio, Alhaalla: Particle Designer sekä Blender). Täysikokoiset kuvat liitteessä 2.

Benchmarking-tutkimuksessa edellä mainittuja neljää editoria (Kuva 2) verrattiin teoria-osuuden listaamien periaatteiden pohjalta. Seuraavassa kappaleessa tulokset käydään läpi suunnitteluperiaate kerrallaan.

4.2 Benchmarking tulokset

Johdonmukaisuus verrattujen editorien toimintojen ja termistön osalta on yleisesti hyvällä tasolla. Suurimpia ongelmakohtia olivat epäselvät toiminnot, joiden vaikutukset eivät olleet riittävän selvästi havainnollistettuja. Erityisesti Visionaire Studiosta löytyy useampia säätöjä, joiden vaikutusta on vaikea ymmärtää ilman dokumentaation tutkimista. Blenderin kohdalla sama ongelma nousi vieläkin häiritsevämmäksi, sillä väärin ymmärretty säätö saattaa aiheuttaa yllättäviä ongelmia, jotka ilmenevät vasta myöhemmässä vaiheessa. Tällöin ongelmakohdan löytäminen saattaa olla jo hyvinkin työläs tehtävä.

Ylimääräiseen tehokkuuteen ei mikään verratuista editoreista panosta erityisen suuresti, joten pikanäppäimiä ja muita tehokäyttäjälle suunnattuja toimintoja käytetään melko vähän. Erityisesti Visionaire Studion tapauksessa niiden tarve voidaan myös kyseenalaistaa, sillä kyseinen editori on lähtökohtaisestikin niin yksinkertainen, että pikanäppäimille sopivia toimintoja on vaikea löytää. Pikanäppäimet tuskin toisivat huomattavaa nopeutusta työtahtiin

tässä tapauksessa. Poikkeus tästä säännöstä on Blender, joka nojaa vahvasti pikanäppäimiin koko yleisen toimintansa puolesta. Tosin tässäkään tapauksessa pikanäppäimillä säästetty aika ei ole huomattavaa, sillä partikkeli-efekteissä suurin osa säädöistä täytyy joka tapauksessa visualisoida muutenkin kuin esikatselussa. Pikanäppäimistä onkin hyötyä lähinnä eri säätösivujen välillä liikkumisessa sekä yleisissä toiminnoissa kuten tilanteen tallentamisessa.

Välitöntä palautetta käyttäjän toimille antavat kaikki editorit ja Blenderiä lukuunottamatta näyttävät myös lopullisen efektin reaaliajassa parametreja muutettaessa. Myös Blender näyttää esikatselun efektistä, mutta lopullisen tuotoksen näkee vasta erillisen mallinnusvaiheen jälkeen. Tämä hidastaa huomattavasti efektin iterointia, mutta kyseisen ohjelman käyttötarkoituksen huomioon ottaessa, tämä on ymmärrettävää. Particle Designer sallii efektin siirtelyn hiirellä, joka helpottaa liikkuvan efektin esikatselua. Unity sallii jopa efektin helpon yhdistämisen peli-objekteihin. Tämä mahdollistaa entistä paremmin efektin arvioimisen lopullisessa kontekstissa, joka puolestaan nopeuttaa iterointia.

Toimintojen ryhmittelyssä eri editorien välillä on huomattavia eroja. Particle Designer jaottelee eri säädöt omiin lokeroihinsa, mutta kaikki säädöt ovat samalla sivulla yhtä aikaa näkyvissä. Toisaalta tämä helpottaa säätöjen vertailua toisiinsa, mutta vaikeuttaa samalla säätökokonaisuuksien hahmottamista. Visionaire Studio sekä Blender jakavat suuren osan säädöistä omille sivuilleen. Visionaire Studion kohdalla tämä toimii loogisesti, mutta Blenderissä osa säädöistä hajaantuu hyvinkin laajalle alueelle. Tämä johtuu siitä, että osa säädöistä ei koske pelkästään partikkeleita, vaan niitä käytetään ohjelmassa muihinkin säätöihin. Toisaalta tämä on hyvä kokeneelle Blender-käyttäjälle, mutta samalla se heikentää partikkeliosuuden käytettävyyttä erillisenä toimintona. Unity järjestelee säädöt omiin moduuleihinsa, joista käyttäjä voi aktivoida vain tarvitsemansa ominaisuudet. Tämä järjestelmä toimii hyvin niin kauan kun eri moduulien välille ei synny konflikteja.

Virhetilanteiden käsittelyssä vakavia ongelmia esiintyi ainoastaan Visionaire Studiolla, jonka tapauksessa koko editori kaatui useamman kerran käytön aikana hävittäen samalla kaikki edellisen tallennuksen jälkeiset muutokset. Tämä ei tietenkään ole hyväksyttävää millekään ohjelmalle. Blenderissä ei yhtä kriittisiä ongelmia löytynyt, mutta partikkelijärjestelmä oli melko helppo saada virheelliseen tilaan. Esimerkiksi vain mallinnusvaiheessa esiintyvien tekstuuri-ongelmien vaatimat toimenpiteet eivät aina olleet helposti hahmotettavissa.

Toimintojen hallinta oli yleisesti hyvällä tasolla hyvin toteutetun välittömän palautteen johdosta. Blenderin hidas efektien iterointi johtaa kuitenkin helposti käyttäjän reagoivaan asemaan, jossa ongelmia korjataan niiden ilmentyessä. Välillä tämä tapahtuu huomattavasti asetuksien säätämisen jälkeen, joka entisestään vahvistaa kontrollin menettämisen tunnetta.

Myös Visionaire Studion kanssa käyttäjä välillä menettää kontrollin tunnetta kun toimintojen vaikutukset eivät ole täysin selviä.

Tietojen saatavuus vaihtelee editoreittain säätöjen ryhmittelyn tapaan. Particle Designer pitää kaikki tiedot kokoajan käyttäjä näkyvillä. Tämä ei kuitenkaan välttämättä helpota tietojen löytämistä, sillä tärkeät tiedot hukkuvat helposti vähemmän tärkeiden tietojen sekaan. Visionaire Studion jaottelu toimii hyvin tässä tapauksessa. Tosin on huomioitava, että Visionaire Studiossa säätöjä on huomattavasti vähemmän kuin muissa editoreissa, joten suurempi määrä säätöjä ei todennäköisesti jakautuisi enää yhtä hyvin omille sivuilleen. Unityn tapa moduloida säädöt toimii hyvin, mutta välillä tiedot saattavat johtaa harhaan kun myöhempi moduuli ylikirjoittaa toisen säädön. Esimerkiksi moduuli, joka säätää partikkelin väriä sen elinkaaren mukaan tekee partikkelin varsinaisesta väriasetuksesta toimettoman. Blender on tässäkin kohtaa heikoilla, sillä partikkelijärjestelmälle tärkeät parametrit hukkuvat helposti muihin osuuksiin. Kokeneelle käyttäjälle tämä ongelma on pienempi, mutta partikkeleihin keskittyessä tietoja joutuu silti keräämään monesta ei paikasta.

Yksittäisten toimintojen peruuttaminen löytyy kaikista editoreista Particle Designer:iä lukuun ottamatta. Tosin Blenderin tapauksessa hidas iterointi tekee tästä toiminnosta vähemmän hyödyllisen, sillä iteraatioiden välillä säätöjä tehdään yleensä useampia kerralla. Kun ongelma lopulta ilmenee, on virheellisen säädön jälkeen todennäköisesti jo tehty muitakin säätöjä, joten yksittäisten säätöjen peruuttamisesta ei ole suurempaa hyötyä.

Dokumentaation tasossa eri editoreilla oli huomattavia eroja. Particle Designerin säädöt ovat suurelta osalta selkeitä, mutta dokumentaatio on kaikin puolin heikkoa tai olematonta. Visionaire Studio sisältää hyvän dokumentaation, jossa selitetään pelkkien säätöjen lisäksi myös partikkelijärjestelmien peruskonsepteja. Sekä Unity että Blender sisältävät hyvän dokumentaation editorien toiminnoista, mutta näillä editoreilla on puolellaan myös laaja käyttäjäkunta. Tämän johdosta erilaisia esimerkkejä ja tutoriaaleja on hyvin helppo löytää.

Vaikka jokainen näistä editoreista on hieman eri tarkoitukseen kehitetty, niin yleisesti ottaen suuri osa käyttöliittymien ratkaisuista ovat keskenään samankaltaisia ja suurimmille eroille on helppo löytää perustellut syyt. Esimerkiksi Blenderin muita heikompi välittömän palautteen antaminen on ymmärrettävää kyseisen ohjelman käyttötarkoituksen vuoksi, sillä mallinnusohjelman päämääränä ei ole efektien reaaliaikainen toistaminen. Taulukko 3 listaa tulokset tiivistetyssä muodossa.

Periaate	Particle Designer	Visionaire Studio	Unity	Blender
Johdonmukaisuus	+	-	+	-
Tehokkuus	+/-	+/-	+/-	+
Välitön palaute	+	+/-	+	-
Toimintojen ryhmittely	-	+/-	+	+/-
Virheiden käsittely	+	-	+	+/-
Toimintojen hallinta	+	+/-	+	-
Tiedon saatavuus	+/-	+	+/-	-
Toimintojen peruuttaminen	-	+	+	+
Dokumentaatio	-	+	+	+/-

Taulukko 3: Tiivistelmä verrattujen editorien vahvuuksista ja heikkouksista

4.3 Haastatteluiden suunnittelu

Haastattelut toteutettiin teemahaastatteluina. Näiden haastatteluiden päämääränä oli ottaa loppukäyttäjä mukaan suunnitteluprosessiin mahdollisimman aikaisin ja antaa mahdollisuus vaikuttaa myös perusratkaisuihin jo editorin suunnitteluvaiheessa. Haastattelun kohteina toimivat kaksi Game Artistia sekä yksi Senior Game Artist, joita kaikkia haastateltiin erikseen. Haastateltavista kaksi olivat haastatteluhetkellä Angry Birds Classic:in graafikkoja ja yksi toimi pitkään samassa projektissa aikaisemmin. Haastateltavat valittiin niin, että jokaisella kohteella olisi hyvä näkyvyys projektin tarpeisiin.

Haastatteluihin valittiin kolme yhteistä pääteemaa. Ensimmäinen teema koski haastateltavien aiempia hyviä ja huonoja kokemuksia muista vastaavista editoreista, jonka avulla pyrittiin saamaan vielä uusia ideoita ja näkökulmia editorin toteutukseen. Toinen teema käsitteli uuteen editoriin kohdistuneita toiveita ja odotuksia antaen haastateltaville mahdollisuuden esittää mielipiteitä editorin toteutuksesta. Kolmantena teemana oli käyttöliittymän ja säätimien intuitiivisuus, jossa pyrittiin vielä kartoittamaan tulevien käyttäjien näkemyksiä hyvästä käyttöliittymästä.

Kokonaisuutena näillä teemoilla pyrittiin saamaan käsitys käyttäjien kokemustasosta partikkelien kanssa työskentelystä sekä kartoittamaan millaiset ratkaisut toimisivat parhaiten kyseisten henkilöiden kohdalla. Haastatteluiden tulokset käydään läpi seuraavassa kappaleessa.

4.4 Haastatteluiden tulokset

Haastattelujen tulokset olivat pitkälti linjassa keskenään, joten yleiset näkemykset ja toiveet olivat hyvin samankaltaisia. Nykyinen järjestelmä koettiin riittämättömäksi tehokkaaseen työskentelyyn ja tulevalta editorilta toivottiin virtaviivaisuutta sekä mahdollisuutta efektien hallintaan mahdollisimman visuaalisella tavalla. Samankaltaiset positiiviset ja negatiiviset kokemukset nousivat esiin aiemmin käytetyistä editoreista riippumatta.

Nykyisen järjestelmän suurimmiksi heikkouksiksi nähtiin erityisesti työskentelyn hitaus sekä osittainen artistisen määrittelyn siirtyminen ohjelmoijan tehtäväksi. Efektien sanallinen selittäminen koettiin liian vaikeaksi ja "rikkinäinen puhelin"-efektin huomioitiin helposti vaarantavan partikkeli-efektien laadun. Ongelman kiertämiseen ehdotettiin efektien esimallintamista erillisessä ohjelmassa, jolloin ohjelmoijalle voitaisiin antaa referenssi-materiaalia. Tämän todettiin kuitenkin vievän huomattavasti ylimääräistä aikaa. Tällaisessa prosessissa myös tärkeät yksityiskohdat sekä herkkää balansointia ja tarkkaa ajoitusta vaativat osuudet voisivat helposti vääristyä. Kaikki tämä yhdistettynä hitaaseen iteraatioon johtavat ympäristöön, jossa täydellisyyden tavoittelu koettiin turhauttavana.

Aiemmista kokemuksista positiivisena esimerkkinä nousi esiin tämän työn benchmarking-osuudessakin hyvin toimivaksi todettu Unity. Sen kyky näyttää efektit reaaliajassa ja oikeassa kontekstissa ovat avainasemassa tehokkaan ympäristön luomisessa. Aiemmat kokemukset muista editoreista osoittivat yleisestä toimivuudestaan huolimatta useita puutteita muun muassa liian hitaan iteroinnin sekä säätöjen epämääräisyyteen suhteen. Näissä tapauksissa arvioitiin jo pelkällä käyttöliittymän hiomisella saavutettavan huomattavaa parannusta käyttökokemukseen.

Toiveiden kerääminen tulevan editorin osalta osoittautui hankalaksi, sillä kellään haastateltavista ei ymmärrettävästi ollut vahvaa näkemystä nykyisen ohjelmointipainotteisen järjestelmän ominaisuuksista ja rajoitteista. Siitä johtuen toiveet liittyivät suurelta osalta itse partikkelijärjestelmään, jonka laajamittainen kehittäminen ei kuitenkaan ollut osa tämän projektin tavoitteita. Tämä kuitenkin osoitti nykyisen järjestelmän puutteita, joita varmasti otetaan huomioon järjestelmän jatkokehityksessä. Näiden muutosten ja lisäysten mahdollistaminen tulevaisuudessa on kuitenkin myös editorin vastuulla, joten laajennettavuuden tärkeys painottui myös haastatteluiden tuloksissa.

Varsinaisesti editoriin liittyvistä toiveista päällimmäisiksi nousivat efektin näkeminen mahdollisimman oikeassa kontekstissa, säätimien helppokäyttöisyyden ja tarkkuuden tasapainottaminen sekä yleinen editorin asettelu, jotta kaikki elementit ovat mahdollisimman helposti saatavilla. Säätimien helppokäyttöisyyden ja tarkkuuden tasapainottaminen tarkoitti

tässä tapauksessa sitä, että yleisesti säätöjen tulisi olla nopeasti vaihdettavissa visuaalisilla apuvälineillä, kuten liukusäätimillä, mutta tarvittaessa arvot tulisi pystyä asettamaan myös täysin tarkasti. Editorin yleinen asettelu on yksi kynnyskysymyksistä käyttökokemuksen määrittämisessä ja partikkeli-editorin yhteydessä tämä onkin haaste tarvittavien säätimien suuren määrän johdosta. Unityn modulaarinen rakenne nähtiin kelvollisena ratkaisuna tähän ongelmaan, mutta moduulien määrän kasvaessa tämänkään rakenteen ei koettu takaavan tehokasta liikkumista säätöjen välillä.

Erilaisten säätimien intuitiivisuudesta keskusteltaessa, näkemykset olivat hyvin yhtenäisiä. Kulmien määrittelyssä kellotaulu viisareineen oli hyväksytyin vaihtoehto, jonka lisäksi kulmien ja muiden koko-suhteiden visualisointiin toivottiin kiinnitettävän huomiota. Pikanäppäimet eivät olleet kovinkaan korkealla toivelistalla, vaan ennen kaikkea visuaalisten säätimien toivottiin olevan mahdollisimman intuitiivisia.

Haastattelut pääosin vahvistivat valittujen käytettävyyden periaatteiden oikeellisuuden. Lisäksi säätimien intuitiivisuus osoittautui odotusten mukaisesti hyvin tärkeäksi. Varsinaisen partikkelijärjestelmän erottelu editorin toiminnoista osoittautui hankalaksi, sillä graafikoilla ei ollut riittävää näkyvyyttä nykyisen järjestelmän toimintaan.

5 Editorin toteutus

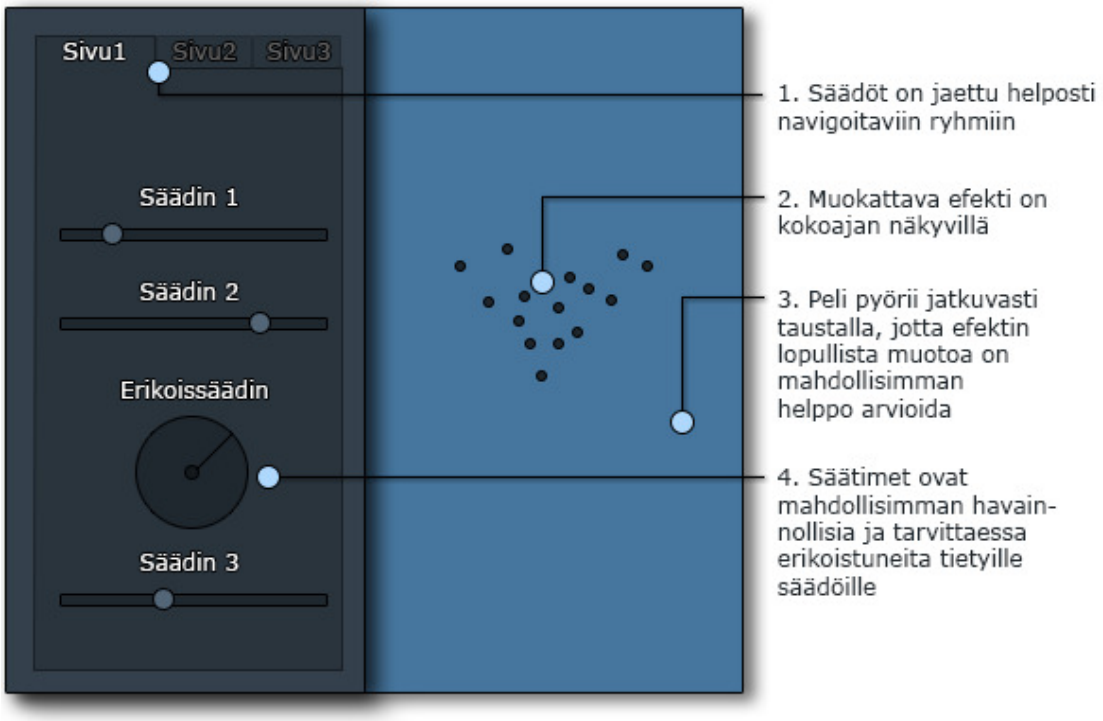
Tässä kappaleessa käydään läpi editorin toteutuksen vaiheet. Aluksi kerätyn tiedon pohjalta koottiin suunnitelma, joka kattaa editorin perusasettelun sekä tärkeimmät toiminnot. Tämän jälkeen kappaleessa 5.2 kartoitetaan lopullinen toteutus teoriaosuuden periaatteiden pohjalta. Lopuksi käydään vielä läpi toteutuksessa kohdattuja haasteita.

Toteutusvaihe aloitettiin luomalla pohjamalli benchmarking- sekä haastatteluvaiheiden tulosten pohjalta. Tätä mallia kehitettiin vielä eteenpäin kappaleessa 3 läpi käytyjen periaatteiden pohjalta. Editori päätettiin toteuttaa pelin sisäisesti erillisen ohjelmiston sijasta, jotta olemassa olevaa järjestelmää voitaisiin hyödyntää mahdollisimman paljon. Samalla mahdollistettiin efektien näyttäminen oikeassa kontekstissa pelin päällä sekä mahdollisuus nähdä reaaliajassa myös muutokset jo valmiiksi pelissä käytössä olevissa efekteissä.

5.1 Suunnitelma

Editoria lähdettiin suunnittelemaan aiemmissa vaiheissa kerätyn tiedon perusteella. Tärkeimmiksi aiheiksi haastatteluiden pohjalta nousivat säätimien visuaalisuus, efektin oikea

konteksti, välitön palaute sekä säätöjen ryhmittely. Kuva 3 näyttää pohjapiirustuksen suunnitellulle editorille, jossa näitä aiheita on käsitelty.

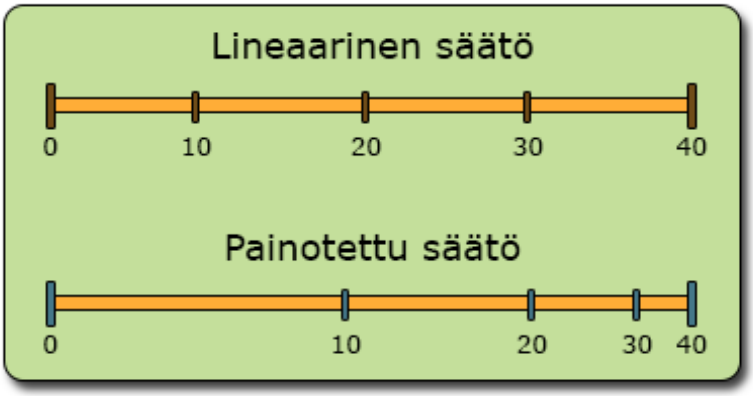


Kuva 3 Suunniteltu pohjamalli toteutettavalle editorille

Välitön palaute säädöille sekä oikea konteksti olivat editorin ehdottomia kulmakiviä, joten ne ohjasivat pitkälti editorin perus asettelua. Efektin tuli olla kokoajan näkyvillä ja sille täytyi varata riittävästi tilaa, joten ruudun jakaminen kahteen osaan vaikutti luonnolliselta ratkaisulta. Oikea puoli ruudusta varattiin efektin näyttämiselle ja samalla kun peli saa pyöriä taustalla, pysyy efektin konteksti myös mahdollisimman todenmukaisena. Efektin havainnointia oikeassa kontekstissa haluttiin helpottaa vielä mahdollisuudella liikutella efektiä vapaasti ympäri ruutua, jolloin liikkuva tai tietyssä osassa ruutua esiintyvä efekti voidaan nähdä mahdollisimman oikeassa kontekstissa. Vasen puoli ruudusta varattiin säätimille. Mukaan haluttiin kuitenkin myös mahdollisuus liikuttaa efektiä vapaasti vasemmalle puolelle ruutua, joten säätöpaneelille lisättiin vaatimus mahdollisuudesta piilottaa se jättäen efekti kuitenkin ruudulle näkyviin.

Säätöjen intuitiiviseen ryhmittelyyn haluttiin myös panostaa ja samalla hyödyntää Unityn kaltaista modulaarista rakennetta, joka samalla helpottaisi editorin myöhempää jatkokehitystä ja laajentamista. Koska ylimääräisten käyttöliittymä-elementtien kehitykseen käytetty aika haluttiin minimoida, säädöt jaettiin yksinkertaisiin ryhmiin ja editorin paneeli jaettiin moduulimaisiin sivuihin. Unityn ratkaisussa yksittäisessä moduulissa saattaa olla vain yksi säätö, mutta käyttöliittymän yksinkertaistamiseksi lähdettiin liikkeelle siitä, että jokaisella sivulla on aina kokonainen ryhmä säätöjä.

Säätimien visualisoinnissa painotettiin erityisesti haastatteluissa ilmenneitä toiveita. Suurin osa parametreista sopivat liukusäätimellä säädettäviksi. Useassa tapauksessa säätöalue on kuitenkin hyvin laaja, vaikka suurin osa säädöistä asettuu pienelle alueelle. Esimerkiksi partikkelien elinaika on suurimmalla osalla efekteistä alle kolme sekuntia, mutta tietyissä tapauksissa tarvitaan jopa 45 sekuntia eläviä partikkeleita. Tämän vuoksi liukusäätimiin kaivattiin painotusta niin, että pienemmissä arvoissa sama säätimen liike vastaa pienempää muutosta kuin suuremmissa arvoissa (Kuva 4). Näin säilytetään pienempien arvojen tarkkuus, mutta sallitaan myös huomattavasti suurempien arvojen säätäminen. Suurempien arvojen voidaan olettaa vaativan vähemmän tarkkuutta kuin asteikoin pienimpien arvojen. Lisäksi kaivattiin myös mahdollisuutta asettaa arvot suoraan kirjoittamalla, jolloin saadaan tarvittaessa asetettua säätimelle tarkka arvo.



Kuva 4 Painotetun säätimen toimintaperiaate

Muista säätimistä kulmien säädöt olivat tärkeimpiä saada intuitiivisiksi. Haastatteluissa mainittu kellotaulu vaikutti hyvältä ratkaisulta, johon vaihtelevaisuussäädön liittämällä saadaan visuaalinen säädin kulmien asettamiselle. Lisäksi kulmien konkreettinen visualisoiminen efektin yhteydessä oli tärkeää, sillä muuten kulman todellinen vaikutus voi jäädä hieman epäselväksi.

Useiden editorien käyttämät käyrät antavat visuaalisen tavan hallita parametreja partikkelien elinaikana, mutta tämä ominaisuus päätettiin jättää mahdolliseksi jatkokehityksen kohteeksi. Alla oleva järjestelmä ei valmiiksi tukenut tällaisia parametreja ja itse säätimen kehittäminen olisi vienyt huomattavasti aikaa jo valmiiksi tiukasta aikataulusta.

Muita pienempiä suunniteltuja ominaisuuksia olivat muun muassa helppokäyttöinen toimintojen peruuttaminen, uusien efektien luonti vanhojen pohjalle ja säätöjen jakaminen pohja-arvo/vaihtelevaisuus säätimiin. Lopullisen toteutuksen ominaisuudet riippuivat

kuitenkin työn etenemisvauhdista verrattuna aikatauluun. Seuraavassa kappaleessa käydään läpi toteutetun editorin toiminnot.

5.2 Toteutettu käyttöliittymä

Käyttöliittymä toteutettiin edellisessä kappaleessa kuvaillun suunnitelman mukaisesti. Tässä kappaleessa tehty toteutus käydään läpi aiemmin määriteltyjen käytettävyyden periaatteiden mukaan. Osa suunnitelluista toiminnoista jouduttiin kuitenkin siirtämään myöhempiin päivityksiin ja nämä tapaukset on käyty läpi seuraavassa kappaleessa. Kuva 5 näyttää lopullisen editorin toiminnassa.

Kuva 5 Toteutettu editori

Toimintojen johdonmukaisuutta pyrittiin parantamaan yhdistämällä säädöt pareiksi, joissa säädöllä on pohja-arvo sekä vaihtelevaisuus arvo (Kuva 6). Täten esimerkiksi partikkelin lähtönopeuden ollessa 20 ja lähtönopeuden vaihtelevaisuuden 10, on partikkelien varsinainen lähtönopeus jotain 15:n ja 25:n väliltä. Alla oleva partikkelijärjestelmä käyttää minimi ja maksimi arvoja, jolloin vastaava tulos saataisiin aikaan asettamalla minimi tasolle 15 ja maksimi tasolle 25. Ensimmäinen iteraatio säätimistä käyttikin tätä järjestelmää suoraan, mutta se osoittautui vähemmän havainnolliseksi kuin pohja-arvo ja vaihtelevaisuus yhdistelmä.

Kuva 6 Kulman säätöparin toimintaperiaate. Visualisointiapuna punaiset linjat havainnollistavat käyttäjälle säätöjen konkreettisen vaikutuksen.

Tehokäyttäjille suunnatut toiminnot osoittautuivat sekä benchmarking- että haastatteluvaiheessa arvioitua vähemmän tärkeiksi. Tämän johdosta niille ei asetettu toteutusvaiheessa korkeaa prioriteettia ja pikanäppäimiä päädyttiinkin tarjoamaan vain globaaleille toiminnoille, jotka ovat käytettävissä kaikissa editorin näkymissä. Tällaisia toimintoja ovat muun muassa editorin käyttöliittymän piilottaminen sekä apupiirtotoimintojen näyttäminen.

Välittömän palautteen tarjoaminen käyttäjälle oli tärkeää toivotun lopputuloksen saavuttamiseksi. Ensimmäinen toimi tämän tavoitteen saavuttamiseksi oli valinta tehdä editori pelin sisäisenä osana erillisen ohjelmiston sijasta ja pitää muokattava efekti kokoajan ruudulla. Tämän johdosta efekti voidaan näyttää mahdollisimman oikeassa kontekstissa reaaliajassa. Tämä periaate myös ohjasi säätimien suunnittelua, sillä säätöjen vaikutusten tuli näkyä jo säätöä tehdessä, eikä vasta arvon valinnan jälkeen.

Toimintojen ryhmittelyssä pyrittiin jakamaan säädöt käytännöllisiin ryhmiin, joista tarvittavat säädöt olisi helppo löytää. Samalla kuitenkin alla olevan partikkelijärjestelmän hierarkiaa pyrittiin noudattamaan, jotta mahdolliset muutokset olisi helppo integroida editoriin. Alla olevaan taulukkoon (Taulukko 4) on listattu valikoituneet ryhmät sekä niiden sisältämät säädöt.

Ryhmän nimi	Kuvaus	Tärkeimmät säädöt
Base	Efektien perussäädöt	Uuden efektin luominen, Efektien poistaminen, Muutosten tallentaminen
Sprites	Efektin käyttämien kuvien asetukset	Kuvien valinta, Animaatio asetukset
Type	Partikkelien suuntauksen määritykset	Partikkelien suuntaus, Partikkelien lähtönopeus
Life	Partikkelin elinkaareen vaikuttavat säädöt	Partikkelien elinaika, Partikkelien pyörimisnopeus, Painovoiman vaikutus
Size	Partikkelien koon määritykset	Partikkelien koko elinkaaren alussa, Partikkelien koko elinkaaren lopussa
Reference	Tietoja, joita käytetään partikkeleita lähetettäessä	Partikkelien määrä, Partikkelien lähetystahti, Lähetysalueen määritykset

Taulukko 4: Editorin säätöjen ryhmittely

Ryhmittelyssä haluttiin ottaa huomioon myös tulevaisuuden tarpeet editorin laajentamiselle, joten jokainen ryhmä ja sen säätimet on määritelty erillisessä tiedostossa. Uuden säätösivun lisääminen vaatii pienimmillään vain uuden tiedoston lisäämistä, joka määrittelee millaisia säätimiä kyseisellä sivulla on ja mitä partikkeli-efektin parametreja ne muokkaavat. Näin varsinaiseen editorin toteutukseen ei tarvitse koskea, jos tarve on vain uuden ominaisuuden lisäämiselle. Reference-sivu (kuva 7) on hyvä esimerkki editorin laajennettavuudesta, sillä se sisältää parametreja joita ei ole aiemmin tallennettu efektin määrittelyyn. Tästä huolimatta sen toiminnot on muiden sivujen tapaan määritelty kokonaan omassa tiedostossaan koskematta varsinaisen editorin toteutukseen.

Kuva 7 Reference-sivu. Oikealla puolella näkyvissä emitter-alueen visualisointia.

Virhetilanteiden käsittelyyn panostettiin huomattavasti alusta alkaen. Osittain virhetilanteiden mahdollisuutta pyrittiin rajoittamaan sillä, että alla olevaa partikkelijärjestelmää pyrittiin hyödyntämään mahdollisimman paljon nykyisessä muodossaan. Tällä tavalla vältetään uusien muutosten aiheuttamat ongelmat ja itse editori pystyy keskittymään omien virhetilanteiden hoitamiseen.

Toimintojen hallinnan vahvistaminen pyrittiin varmistamaan yksinkertaisen käyttöliittymän sekä vahvan välittömän palautteen avulla. Näiden osuuksien toimiessa hyvin käyttäjälle jää tunne siitä, että hän on ollut toimintojen hallinnassa eikä ole joutunut reagoivaan rooliin työtä tehdessään.

Hyvä tiedon saatavuus pyrittiin varmistamaan paitsi panostamalla säätöjen ryhmittelyyn, myös visualisoimalla tärkeimpiä tietoja efektin yhteydessä. Avustavat piirtotoiminnot, jotka visualisoivat säätöjä myös efektin yhteydessä vähentävät tarvetta säätöjen varsinaisten arvojen tarkistelemista.

Editorin käyttöönottoa sekä itse käyttöä selvennettiin erillisellä dokumentilla. Käyttöönoton vaatimia toimenpiteitä pyrittiin minimoimaan, mutta jäljelle jääneet pakolliset toimet käydään läpi tässä dokumentissa. Lisäksi kyseinen dokumentti käy läpi kaikki editorin säädöt selittäen niiden toiminnot. Koska editorin laajennettavuus oli myös yksi tavoitteista, on tämä dokumentti asetettu kaikkien käyttäjien saataville siten, että kuka vain editoriin muutoksia tekevä voi tarvittaessa päivittää myös dokumentaatiota.

5.3 Kohdatut haasteet

Toteutusvaiheen rajoitettu aikataulu aiheutti joidenkin suunniteltujen ominaisuuksien implementaation lykkäämistä myöhempään ajankohtaan. Näistä merkittävimpänä toimintojen peruuttaminen toteutettiin kompromissina niin, että kaikkien muutosten peruuttaminen kerralla onnistuu, mutta yksittäisten muutosten ei. Mahdollinen toteutus yksittäisten muutosten peruuttamiselle ehdittiin kuitenkin jo testata, mutta kyseinen ominaisuus vaatisi silti vielä työtä.

Myöskään alla olevan partikkelijärjestelmän kanssa ei päästy täydelliseen yhteensopivuuteen ilman muutoksia. Alkuperäisen suunnitelman mukaan editorin ei ollut tarkoitus vaatia mitään muutoksia partikkeli-efektien organisointiin, mutta toimivan versiohallinnan varmistamiseksi efektit jouduttiin jakamaan yhdestä keskustiedostosta omiin tiedostoihinsa. Tällä ei kuitenkaan ole vaikutusta muuhun kuin partikkelitietojen lataamisprosessiin, joten seuraukset varsinaiselle pelikehitykselle jäävät pieniksi.

Vaikka valmis partikkelijärjestelmä antoi valmiit puitteet työskentelylle, asetti se samalla kuitenkin myös rajoitteita. Luonnolliset ryhmittelyjen löytäminen osoittautui osittain hankalaksi, sillä alla olevasta logiikasta ei tahdottu ajautua liian kauas. Tällainen liiallinen irtautuminen olisi voinut hankaloittaa editorin jatkokehitystä, jos näitä vanhempia toimintoja haluttaisiin muuttaa myöhemmin. Lopullinen ryhmittely osoittautui kuitenkin riittävän toimivaksi, vaikka se ei aivan optimaalinen lopulta ollutkaan. Ryhmittelyssä selkein ratkaisu olisi saada kaikki esimerkiksi partikkelien liikerataan vaikuttavat säädöt samalle sivulle, joka tässä tapauksessa osoittautui kuitenkin hankalaksi. Tämä johtui siitä, että lähtönopeuden ja suunnan säädöt olivat partikkelijärjestelmässä oma erillinen kokonaisuutensa, joka editorissa ryhmittyi loogisemmin omaksi sivukseen.

5.4 Tulosten arviointi

Projektin tuloksia mitattiin vielä lopuksi antamalla editori loppukäyttäjille testattavaksi ja seuraamalla käyttökokemusta vierestä. Vaikka tulosten arviointi oli tässä vaiheessa vielä melko kevyttä, käyttäjiltä saatu ensimmäinen palaute oli hyvin positiivista. Editorin koettiin tehostavan huomattavasti partikkeli-efektien kanssa työskentelyä, joka oli myös projektin päätavoite.

Oletetusti itse partikkelijärjestelmään toivottiin lisäominaisuuksia, mutta niiden lisääminen ei varsinaisesti kuulunut osaksi tätä projektia. Tämä loppukäyttäjien tekemä testaus kuitenkin osoitti jo editorin hyviä puolia uusien ideoiden muodossa. Partikkelijärjestelmään toivotut muutokset osoittivat, että järjestelmän tullessa tutummaksi graafisesta ilmeestä vastaavat henkilöt pystyvät kommunikoimaan toiveensa entistä tarkemmin partikkelijärjestelmästä vastaaville tahoille.

6 Yhteenveto ja johtopäätökset

Tämän opinnäytetyön tarkoituksena oli tutkia partikkeli-editorien käytettävyyttä sekä kehittää editori Angry Birds Classic -pelin tarpeisiin. Teoriapohjaksi koottiin yhdeksän käytettävyyden periaatetta, joita hyödynnettiin kaikissa projektin vaiheissa. Editorien käytettävyyttä tutkittiin suorittamalla benchmarking-tutkimus neljän partikkeli-editorin kesken sekä selvittämällä loppukäyttäjien tarpeita teemahaastatteluiden avulla. Lopuksi kehitettiin editori kerätyn tiedon pohjalta.

Tutkimusvaiheen tulokset osoittavat, että erityisesti partikkeli-editorien käytettävyydessä kolme tärkeintä aluetta ovat välitön palaute, efektin oikea konteksti sekä säätimien intuitiivisuus. Pahimpia sudenkuoppia taas ovat hitaat iteraatiosyklit sekä epäselvät säädöt.

Nämä tulokset eivät ole kovinkaan yllättäviä, mutta silti useat verratuista editoreista eivät onnistuneet näiden periaatteiden täyttämisessä.

Haastatteluiden sekä saadun palautteen perusteella voidaan todeta, että itse partikkelijärjestelmän jatkokehitys olisi hyödyllistä. Nykyisessä muodossaankin editori kuitenkin tarjoaa käyttäjille mahdollisuuden tutustua nykyiseen partikkelijärjestelmään ja esittää toiveitaan jatkokehityksestä entistä konkreettisemmin termein. Koska editorin laajentaminen on tehty mahdollisimman helpoksi, ei sen tulisi myöskään hidastaa haluttujen toimintojen implementoimista alla olevaan partikkelijärjestelmään. Editorin puolella jatkokehityksenä toimintojen peruuttaminen olisi tärkeää saada käyttöön. Lisäksi esimerkiksi sallimalla parametrien säätämisen käyrillä voitaisiin editorin intuitiivisuutta parantaa entisestään, mutta tämä vaatisi työtä myös partikkelijärjestelmän puolelle.

Jatkotutkimusta aiheeseen voitaisiin kohdistaa esimerkiksi 3D efektien hallintaan, joka asettaa useita uusia vaatimuksia editorille. Muun muassa kontekstin hallinta 3D-ympäristössä on huomattavasti hankalampi toteuttaa, sillä pelkkä efektin piirtäminen pelin päälle ei enää ole riittävä ratkaisu. Lisäksi säätöjen vaikutusten visualisointi efektin yhteydessä on 3D-ympäristössä hankalampaa.

Lähteet

263 million monthly active users in December. Rovio.com 11.1.2013. Viitattu 27.1.2013 http://www.rovio.com/en/news/blog/261/263-million-monthly-active-users-in-december/

Koistinen, O. 2012. Roviolla on jo 500 työntekijää. Helsingin Sanomat 23.11.2012. Viitattu 27.1.2013 http://www.hs.fi/paivanlehti/talous/Roviolla+on+jo+500+ty%C3%B6ntekij%C3%A4%C3%A4/a1353562885256

Lander, J. 1998. The Ocean Spray in Your Face. Game Developer. 7/1998.

Nielsen, J. 1995. 10 Usability Heuristics. 1.1. 1995. Viitattu 3.3.2013 http://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen, J. 2000. Designing Web Usability. New Riders Publishing.

Lappalainen, E. Rovio takoi 152 miljoonan liikevaihdon - Tulos 77 miljoonaa. Talouselämä 3.4.2013. Viitattu 1.5.2013 http://www.talouselama.fi/uutiset/rovio+takoi+152+miljoonan+liikevaihdon++tulos+77+miljoonaa/a2177959

Rovio teki yrityskaupan - osti suomalaisen Futuremarkin pelistudion. Talouselämä 27.3.2012. Viitattu 27.1.2013 http://www.talouselama.fi/uutiset/rovio+teki+yrityskaupan++osti+suomalaisen+futuremarkin+pelistudion/a2093457

Shneiderman, B. 1998. Designing the User Interface. Addison-Wesley.

Vaalisto, H. 2011. Angry Birdsin tekijä osti animaatiostudion. Digitoday 1.6.2011. Viitattu 27.1.2013 http://www.digitoday.fi/viihde/2011/06/01/angry-birdsin-tekija-osti-animaatiostudion/20117797/66

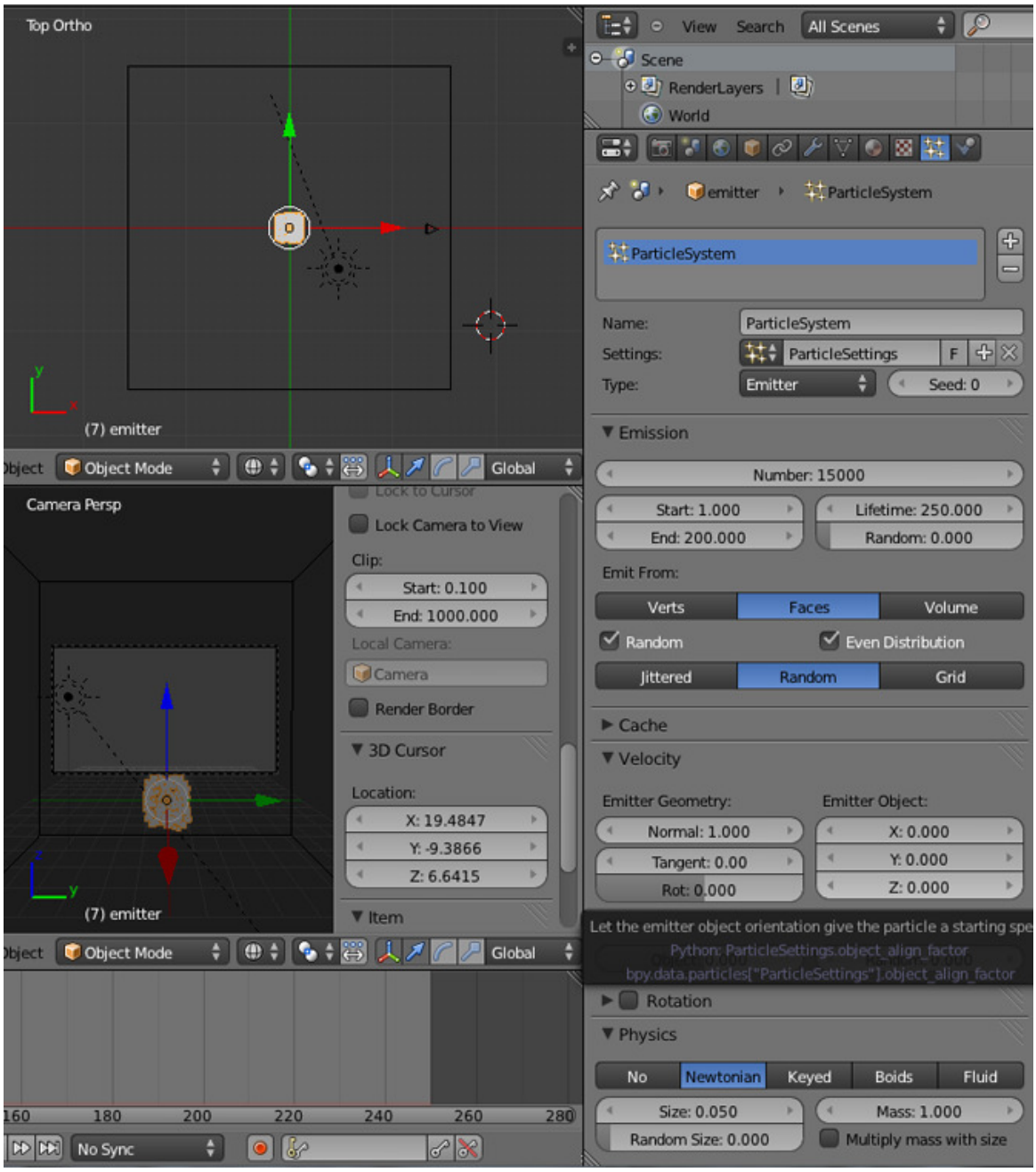
Victor, B. 2012. Learnable programming. Viitattu 2.3.2012. http://worrydream.com/#!/LearnableProgramming

Kuvat

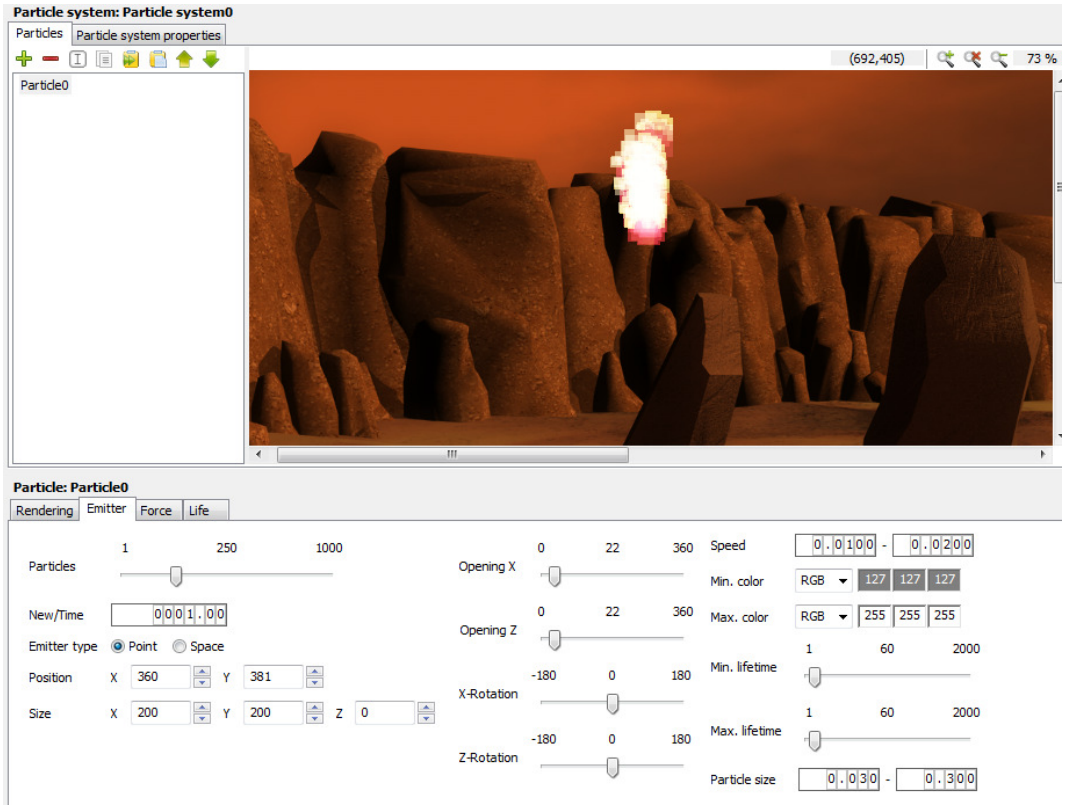
Taulukot

Liite 1: Kuvat käyttöliittymistä

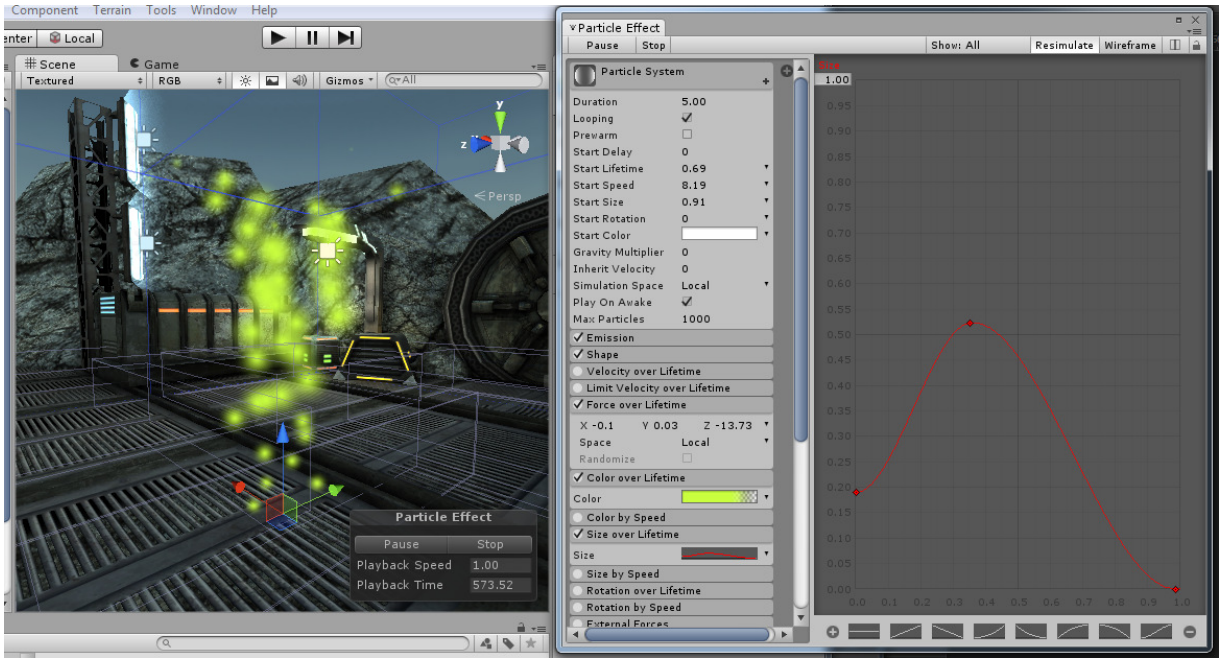
Blender:



Visionaire Studio:



Unity:



Particle Designer: